

Fig.1

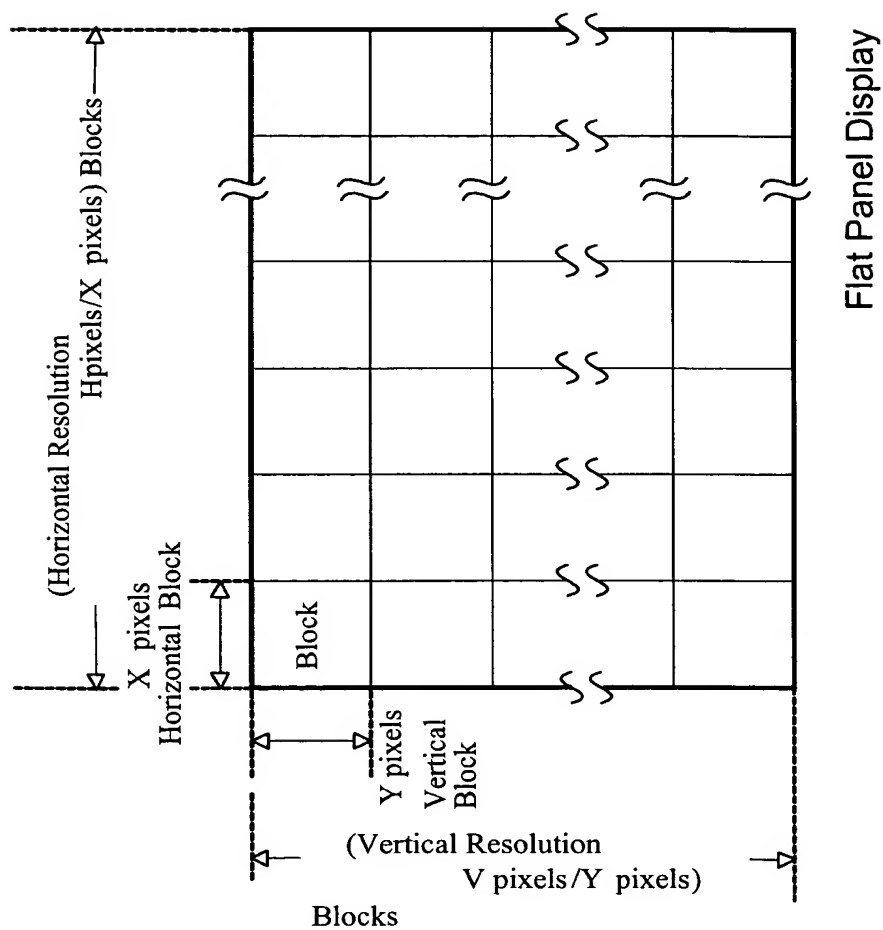


Fig.2

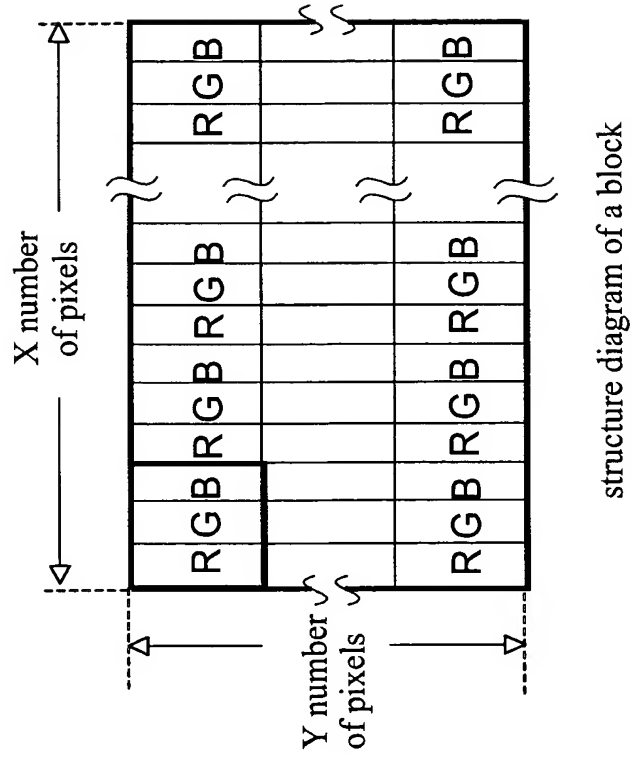


Fig. 3

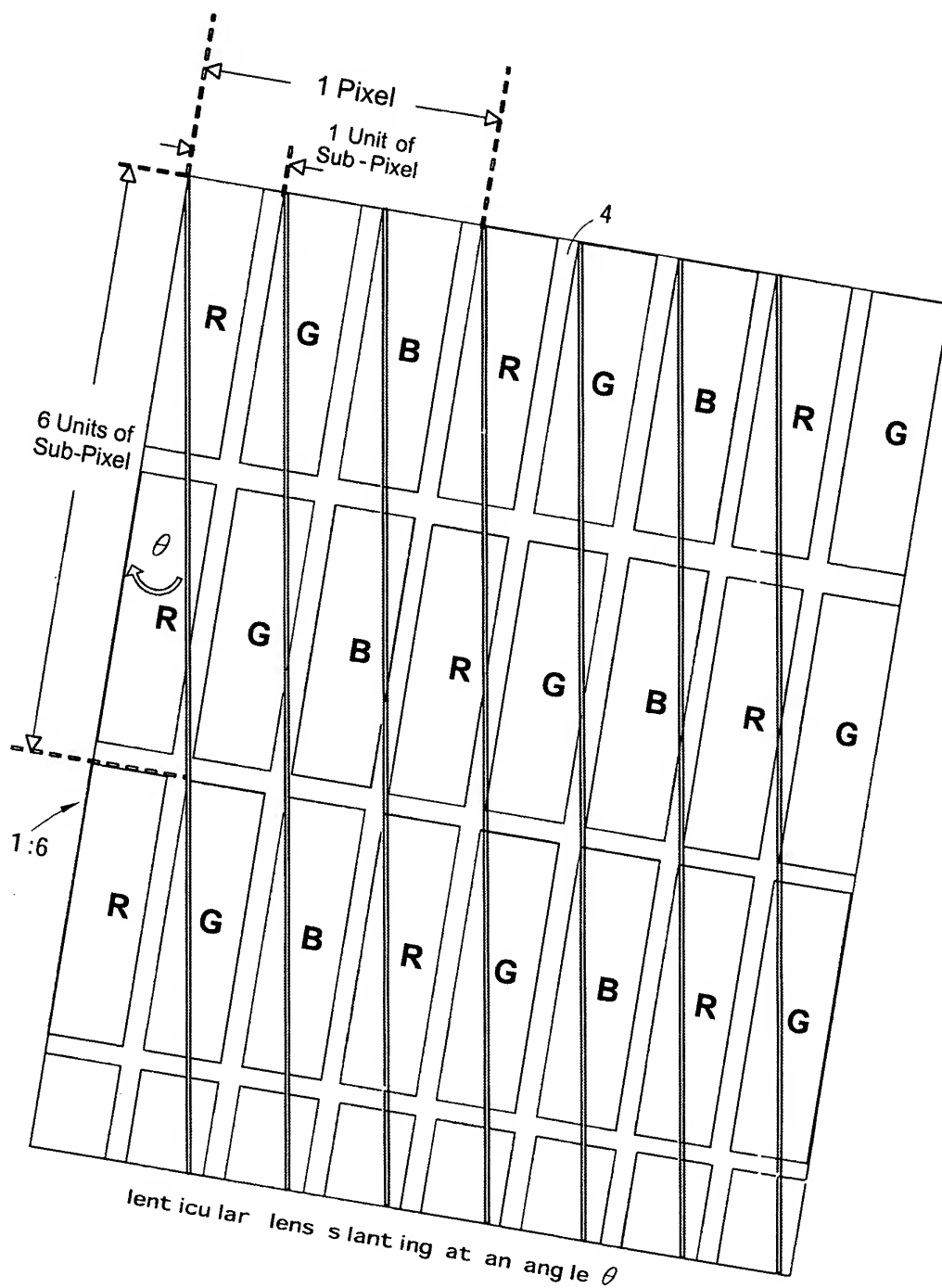


Fig.4

Illustration of Algorithm			
Each view image is separately stored in the block (Total number of pixels : (Block_x) ×(block_y))			
The view image stored in each block will be rearranged by sub - pixels and placed in an appropriate address (i.e., must be in accordance with the specification of lenticular lens, LCD dot pitches, and the number of views).			
The pixels of each view stored in each block (Total number of pixels : (Block_x) × (Block_y)) will be sequentially mapped to the small Map Blocks at different Destinations. (the size is: (N_x)×(N_y) pixels)			
N : (number of views)	8、9、10、12 N = 8    N = 9    N = 10		
N_x : (number of horizontal blocks)	4	3	5
N_y : (number of vertical blocks)	2	3	2
Res_x : (horizontal resolution)	1024、1280、1600, .....		
Res_y : (vertical resolution)	768、1024、1200, .....		
Block_x : (width of each block)	Res_x DIV N_x		
Block_y: (hight of each block)	Res_y DIV N_y		
Block_Start : (the starting address of each block)			
Line_Start : (the starting address of each row block)			
Source(x, y) : (the address of the pixel of the view image inside the block that is to be rearranged)			
Destination(X,Y) : (the pixel Source(x, y) of the view that is going to be rearranged will be rearranged to the address of the small Map Block at its corresponding Destination site)			
Processing Algorithm: For (n = 0 to N-1) { // Process the view image stored in each block in a sequential order // Find the starting address of each block (Block_Start) Block_Start = (n DIV N_y) MUL Block_y MUL Res_x ADD (n MOD N_y) MUL Block_x L_Destination(X, Y) = 0 For (y=1 to Block_y) { // Process each block sequentially in an order of one row block followed by another row block // Find the starting address of each row block (Line_Start) Line_Start = Block_Start ADD (y SUB 1) MUL Res_x			

Fig .5A

Destination(X,Y) = L\_Destination(X,Y)

```

For (x=1 to Block_x)
{
  // Process each row block sequentially in an order of one pixel
  // followed by another pixel
  // Find the pixel point (Source(x, y)) of the view image that is
  // going to be rearranged
  Source(x, y) = Line_Start ADD (x SUB 1)
  // Rearrange the R, G, and B of the pixel point Source(x, y) to their
  // corresponding addresses
  // the R of the Source(x, y) should be rearranged to the R at the
  Destination(X1, Y1)
  // the G of the Source(x, y) should be rearranged to the G at the
  Destination(X2, Y2)
  // the B of the Source(x, y) should be rearranged to the B at the Destination(X3, Y3)
  DestinationRGB(X1, Y1, R) = Source(x,y). R
  DestinationRGB(X2, Y2, G) = Source(x,y). G
  DestinationRGB(X3, Y3, B) = Source(x,y). B

  // Horizontally shift the small Map Block at the Destination site to the address of next
  // small Map Block

  Destination(X, Y) = Destination(X, Y) ADD N_x
}
// Vertically shift the small Map Block point (N_y x N_y)
// to the small Map Block in the next row
L_Destination(X,Y) = L_Destination(X,Y) ADD
(N_y MUL Res_x)
}

```

#### Complementary Illustrations:

DestinationRGB(X, Y, R) is the address of R sub - pixel at the Destination(X,Y)  
 DestinationRGB(X, Y, G) is the address of G sub - pixel at the Destination(X,Y)  
 DestinationRGB(X, Y, B) is the address of B sub - pixel at the Destination(X,Y)  
 DestinationRGB(X, Y) is the address of the pixel point Source(x, y)  
 corresponding to the small Map Block at the Destination site  
 DestinationRGB(X1, Y1, R), DestinationRGB(X1,Y1,G), and  
 DestinationRGB(X1,Y1, B) are the mapping addresses of R, G, B  
 respectively, which are corresponding to the address  
 DestinationRGB(X,Y) of the small Map Block with one offset, wherein  
 the offset is decided by Fn(n, Lx, rgb).  
 DestinationRGB(X1, Y1, R) = DestinationRGB(X, Y) + Fn(n, Lx, R)  
 DestinationRGB(X2, Y2, G) = DestinationRGB(X, Y) + Fn(n, Lx, G)

Fig .5B

wherein  $F_n(n, L_x, rgb)$  is a real time Hash Map Table for calculating the offset of the corresponding small Map Block address.  
 Parameter:  $n$  = number of view; each  $F_n$  has its corresponding Map Table  
 (different Destination Row( $y$ ) will result in different Map Value)  
 $rgb = R \text{ or } G \text{ or } B$

for example  $N=8$

$n = 0 \sim 7$   
 $L_x = 0 \sim 3$   
 $Rgb = R, G, B$   
 $F_n(0, 0, R) = 1$   
 $F_n(0, 0, G) = 5$   
 $F_n(0, 0, B) = 9$   
 $F_n(1, 0, R) = 2 + Res\_x * 3$   
 $F_n(1, 0, G) = 6 + Res\_x * 3$   
 $F_n(1, 0, B) = 10 + Res\_x * 3$   
 $F_n(2, 0, R) = 2$   
 $F_n(2, 0, B) = 10$   
 $F_n(2, 0, G) = 6$   
 $F_n(3, 0, R) = 3 + Res\_x * 3$   
 $F_n(3, 0, G) = 7 + Res\_x * 3$   
 $F_n(3, 0, B) = 11 + Res\_x * 3$   
 $F_n(4, 0, R) = 3$   
 $F_n(4, 0, G) = 7$   
 $F_n(4, 0, B) = 11$   
 $F_n(5, 0, R) = 4 + Res\_x * 3$   
 $F_n(5, 0, G) = 8 + Res\_x * 3$   
 $F_n(5, 0, B) = 12 + Res\_x * 3$   
 $F_n(6, 0, R) = 4$   
 $F_n(6, 0, G) = 8$   
 $F_n(6, 0, B) = 12$   
 $F_n(7, 0, R) = 4 + Res\_x * 3$   
 $F_n(7, 0, G) = 8 + Res\_x * 3$   
 $F_n(7, 0, B) = 12 + Res\_x * 3$

.....  
 .....

Processing algorithm for stereoscopic image synthesizer

F ig .5C